tokenization

TF-IDF

The Dot Product

Let’s begin with the definition of the dot product for two vectors: \vec{a} = (a_1, a_2, a_3, \ldots) and\vec{b} = (b_1, b_2, b_3, \ldots), where a_n and b_n are the components of the vector (features of the document, or TF-IDF values for each word of the document in our example) and the\mathit{n} is the dimension of the vectors:

  \vec{a} \cdot \vec{b} = \sum_{i=1}^n a_ib_i = a_1b_1 + a_2b_2 + \cdots + a_nb_n  

As you can see, the definition of the dot product is a simple multiplication of each component from the both vectors added together. See an example of a dot product for two vectors with 2 dimensions each (2D):

  \vec{a} = (0, 3) \\   \vec{b} = (4, 0) \\   \vec{a} \cdot \vec{b} = 0*4 + 3*0 = 0  

The first thing you probably noticed is that the result of a dot product between two vectors isn’t another vector but a single value, a scalar.

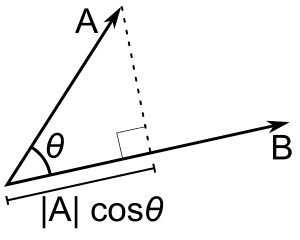
This is all very simple and easy to understand, but what is a dot product? What is the intuitive idea behind it? What does it mean to have a dot product of zero? To understand it, we need to understand what the geometric definition of the dot product is:

  \vec{a} \cdot \vec{b} = \|\vec{a}\|\|\vec{b}\|\cos{\theta}  

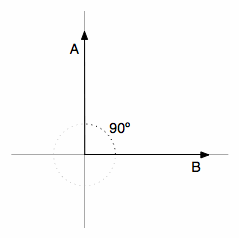
Rearranging the equation to understand it better using the commutative property, we have:

  \vec{a} \cdot \vec{b} = \|\vec{b}\|\|\vec{a}\|\cos{\theta}  

So, what is the term\displaystyle \|\vec{a}\|\cos{\theta}? This term is the projection of the vector \vec{a} into the vector \vec{b} as shown on the image below:

[](http://blog.christianperone.com/wp-content/uploads/2013/09/Dot_Product.png)The projection of the vector A into the vector B. By Wikipedia.

Now, what happens when the vector \vec{a} is orthogonal (with an angle of 90 degrees) to the vector \vec{b} like on the image below?

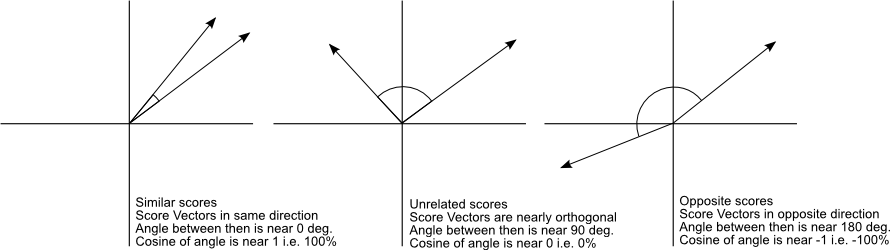
[](http://blog.christianperone.com/wp-content/uploads/2013/09/vectors1.gif)Two orthogonal vectors (with 90 degrees angle).

There will be no adjacent side on the triangle, it will be equivalent to zero, the term \displaystyle \|\vec{a}\|\cos{\theta} will be zero and the resulting multiplication with the magnitude of the vector \vec{b} will also be zero. Now you know that, when the dot product between two different vectors is zero, they are orthogonal to each other (they have an angle of 90 degrees), this is a very neat way to check the orthogonally of different vectors. It is also important to note that we are using 2D examples, but the most amazing fact about it is that we can also calculate angles and similarity between vectors in higher dimensional spaces, and that is why math let us see far than the obvious even when we can’t visualize or imagine what is the angle between two vectors with twelve dimensions for instance.

## The Cosine Similarity

The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them. This metric is a measurement of orientation and not magnitude, it can be seen as a comparison between documents on a normalized space because we’re not taking into the consideration only the magnitude of each word count (tf-idf) of each document, but the angle between the documents. What we have to do to build the cosine similarity equation is to solve the equation of the dot product for the\cos{\theta}:

And that is it, this is the cosine similarity formula. Cosine Similarity will generate a metric that says how related are two documents by looking at the angle instead of magnitude, like in the examples below:

[](http://blog.christianperone.com/wp-content/uploads/2013/09/cosinesimilarityfq1.png)The Cosine Similarity values for different documents, 1 (same direction), 0 (90 deg.), -1 (opposite directions).

Note that even if we had a vector pointing to a point far from another vector, they still could have a small angle and that is the central point on the use of Cosine Similarity, the measurement tends to ignore the higher term count on documents. Suppose we have a document with the word “sky” appearing 200 times and another document with the word “sky” appearing 50, the Euclidean distance between them will be higher but the angle will still be small because they are pointing to the same direction, which is what matters when we are comparing documents.

Now that we have a Vector Space Model of documents (like on the image below) modelled as vectors (with TF-IDF counts) and also have a formula to calculate the similarity between different documents in this space, let’s see now how we do it in practice using [scikit-learn (sklearn)](http://scikit-learn.org/" \o "Scikit-Learn" \t "_blank).

[](http://blog.christianperone.com/wp-content/uploads/2013/09/vector_space.png)Vector Space Model

## Practice Using Scikit-learn (sklearn)

\* In this tutorial I’m using the Python 2.7.5 and Scikit-learn 0.14.1.

The first thing we need to do is to define our set of example documents:

1. documents = (
2. "The sky is blue",
3. "The sun is bright",
4. "The sun in the sky is bright",
5. "We can see the shining sun, the bright sun"
6. )

And then we instantiate the Sklearn TF-IDF Vectorizer and transform our documents into the TF-IDF matrix:

1. **from** sklearn.feature\_extraction.text **import** TfidfVectorizer
2. tfidf\_vectorizer = TfidfVectorizer()
3. tfidf\_matrix = tfidf\_vectorizer.fit\_transform(documents)
4. print tfidf\_matrix.shape
5. (4, 11)

Now we have the TF-IDF matrix (**tfidf\_matrix**) for each document (the number of rows of the matrix) with 11 tf-idf terms (the number of columns from the matrix), we can calculate the Cosine Similarity between the first document (“The sky is blue”) with each of the other documents of the set:

1. **from** sklearn.metrics.pairwise **import** cosine\_similarity
2. cosine\_similarity(tfidf\_matrix[0:1], tfidf\_matrix)
3. array([[ 1. , 0.36651513, 0.52305744, 0.13448867]])

The **tfidf\_matrix[0:1]**is the Scipy operation to get the first row of the sparse matrix and the resulting array is the Cosine Similarity between the first document with all documents in the set. Note that the first value of the array is 1.0 because it is the Cosine Similarity between the first document with itself. Also note that due to the presence of similar words on the third document (“The sun in the sky is bright”), it achieved a better score.

If you want, you can also solve the Cosine Similarity for the angle between vectors:

  \cos{\theta} = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|}  

We only need to isolate the angle (\theta) and move the \cos to the right hand of the equation:

  \theta = \arccos{\frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|}}  

The \arccos is the same as the inverse of the cosine (\cos^-1).

 Lets for instance, check the angle between the first and third documents:

1. **import** math
2. # This was already calculated on the previous step, so we just use the value
3. cos\_sim = 0.52305744
4. angle\_in\_radians = math.acos(cos\_sim)
5. print math.degrees(angle\_in\_radians)
6. 58.462437107432784

And that angle of ~58.5 is the angle between the first and the third document of our document set.

Cosine similarity